June 3, 1999

Human Factors & the Web

Towards Web Macros: a Model and a Prototype System for Automating Common Tasks on the Web

Alex Safonov, Joseph Konstan, and John Carlis

University of Minnesota

Department of Computer Science and Engineering
4-192 EE/CS Bldg., 200 Union St SE

Minneapolis, MN 55455 USA
{safonov,konstan,carlis}@cs.umn.edu

Abstract

The Web is changing from a web of static hypermedia documents to a web of interactive applications. We describe scenarios illustrating user interactions with the Web that represent *automatable tasks*, and show how traditional mechanisms for managing the complexity of the Web - bookmarks - break down in this environment. We introduce Web Macros, the approach for capturing and reusing common automatable tasks, and main requirements for Web Macros: emulating users' navigation and interaction on the Web, parameterization, and authoring by example. We report on our prototype proxy-based system for recording and validating Web macros, and outline directions for future work.

Introduction

Through the last several years, the Web has not only grown in size but also has changed profoundly. In its early years, the Web was primarily a repository of static documents: scientific papers, tutorials, standards, electronic texts, etc. Today, more and more interactive information services, both commercial and free, are becoming available on the Web. Examples of these include electronic newspapers and magazines; online bookstores; computer stores and auctions; airline, hotel, and rental car reservations; driving trip planners; weather updates; digital libraries and bibliography databases, etc. Using these can increase convenience and produce significant savings in time.

With the growth of quantity and diversity of information services on the Web, web user interaction is becoming more complex. Originally, a classic hypertext model of static documents with unidirectional links accurately represented the state of the Web. Over the last several years, the Web has been changing from a *web of documents* to a *web of interactive applications*. A growing number of Web sites invite (and require) a user to interact with them through forms and applets; they rely on dynamically-generated documents, and maintain user and transaction information on the server.

In hypermedia systems preceding the Web and on the Web since its creation, users have been employing various tools and interfaces to manage the wealth of information and aid navigation. Studies have shown that users create *personalized views* of the Web - subspaces of the Web with the logical structure imposed by and familiar to the author[ABC98]. Bookmarks, or favorites, are the primary mechanism for maintaining personalized Web views, and are provided by popular browsers.

Bookmarks, however, have significant limitations in the context of the web of applications. Since bookmarks record only a static URL of a document provided by the server at the time bookmark was created, they cannot capture a document dynamically generated by a CGI script, unless all dynamic information is encoded in the document's URL. More generally, bookmarks cannot capture a *process* of user interaction with the Web - and the process can be required to reproduce a desired result, as we will show in the next section discussing interaction scenarios.

Interaction Scenarios

We are interested in interaction processes where the desired results, and not the intermediate steps, are important. These processes can be considered *automatable Web tasks* - this is in contrast with the research on hypermedia trails and paths. Our interest in automating tasks has influenced our choice of scenarios.

Scenario One: Author Search in a Bibliography Database

The Ovid bibliography search engine provides free Web-based searching services to students and faculty at many U.S. universities, including the University of Minnesota (http://ovid.lib.umn.edu/ovidweb/ovidweb.cgi). Ovid can be searched by keyword, title, and author name. Ovid offers a choice of bibliography databases, whose contents typically differ even though a citation might be present in more than one database. Searching by author name using the Ovid Web interface involves the following steps:

1. Enter the user's X.500 username and password, needed to validate him or her as a University affiliate and to gain access to the Ovid database.

2. Select a database from the list and follow a link to it.

3. Follow the link to the "Author Search" page by clicking on an image.

4. Enter author name and submit the form (Figure 1), retrieving a list of potentially matching authors.

5. Select one or more of the matching authors and retrieve the list of their publications .

6. Review retrieved citations, select a format for saving them, and request the system to display the citation(s) in the selected format. These intermediate (database-specific) search results that must be manually merged after all desired databases have been searched.

7. Re-run the above process for another database by following the "Change Database" link, selecting a desired database from the list, selecting "Yes" and submitting the form (Figure 2.)

8. Repeat for all desired bibliography databases (INSPEC, Compendex, etc.)

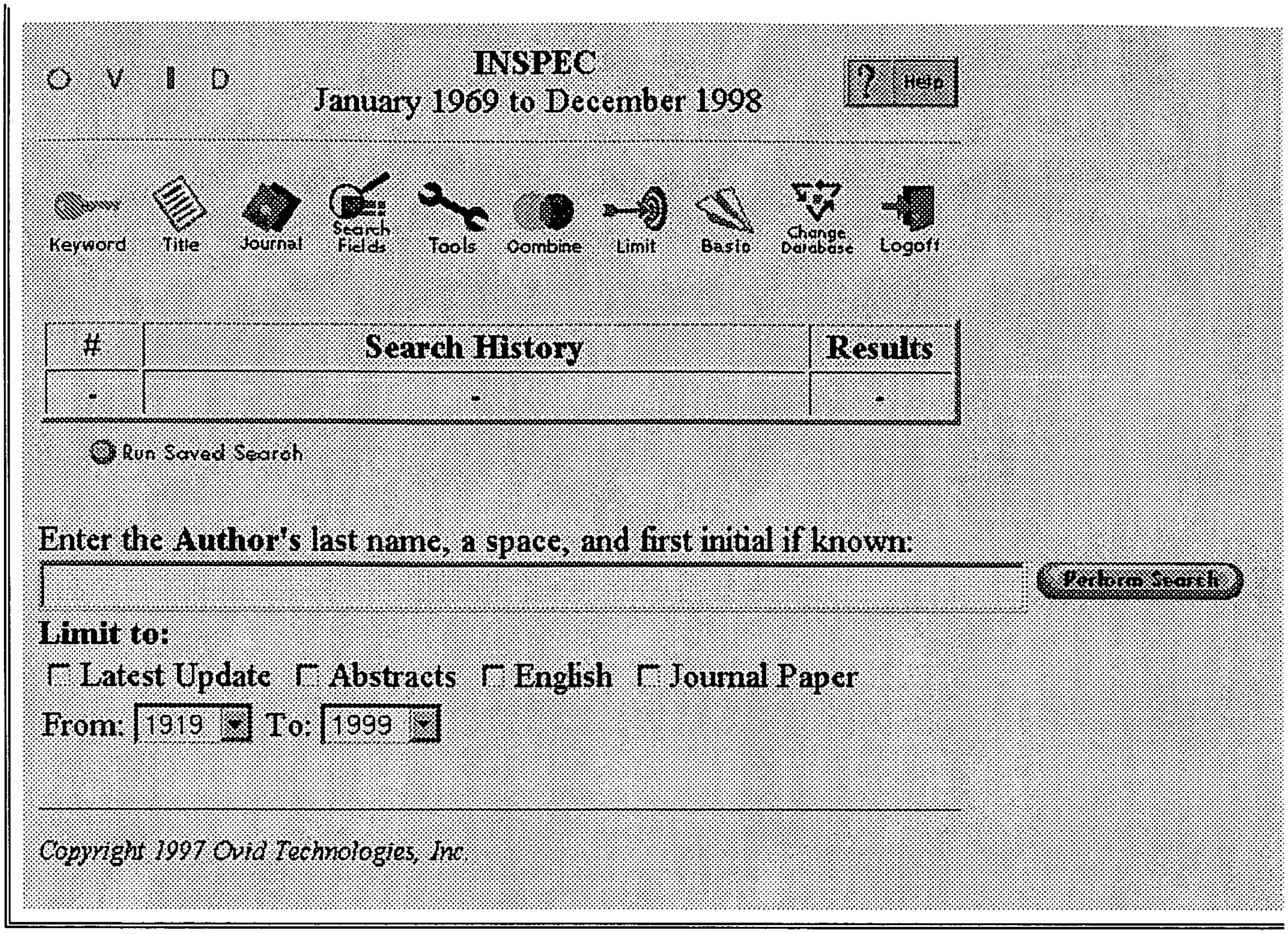

Figure 1. Searching the INSPEC database by author

The database search engine retrieves all authors whose names are match the specified one. The user needs to make a cognitive effort to select an author from the list of matches. This is likely to occur when she runs a search for a specific author's publications for the first time. Even in this exploratory mode, some steps, such as logging in to the Ovid database, selecting Author Search, and saving citations in a desired format, are the same for all searches.

Consider a repeat search, when the user provides enough information to uniquely identify an author, and wants to retrieve all publications by this author. In this case, all steps in the scenario above can be automated, and the only parameter is the author name to search on. It is the process of interaction producing a downloadable file or a page with search results that must be captured if the user wants to add it to her personalized view of the Web as a common Web task. With bookmarks, only the initial step of the process - the Ovid login page - can be captured.

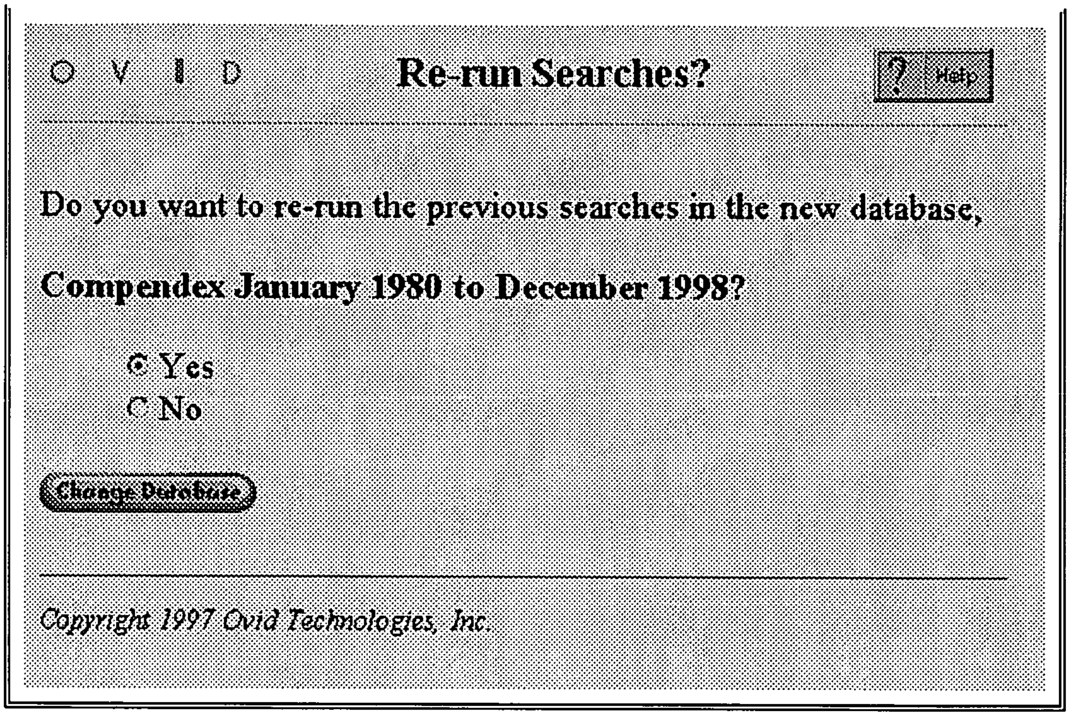

Figure 2. Repeating author search on a different bibliography database

Scenario Two: Persistent and Shareable Shopping Cart

Many on-line stores, such as Bunta Technology (http://www.bunta.com/), provide shopping carts where a customer can place items before buying them. Fewer stores support persistence of shopping carts between sessions, or passing shopping carts to other users for review and purchasing. However, a person creating a shopping cart and a person actually making the purchase can be different. Consider the case of buying computer components. In an office environment, an employee responsible for upgrading computers may need to create a shopping cart with selected components and pass it to his manager for purchase authorization. In a home environment, an expert computer user may assist a friend by selecting the optimal configuration and placing items in the shopping cart, but leaving the final decision to buy to the other person. To support persistent and shareable shopping carts, the process of populating a cart must be captured. In addition, information the server uses to establish user identity (cookies) must be associated with this process.

Other scenarios that describe automatable Web tasks include book searches accessing multiple libraries; connecting Ovid search results with a library search; checking fares at multiple reservation services; and automated bidding in on-line auctions.

Web Macros: a Solution Approach

The analysis of multiple interaction scenarios, including those outlined in the previous section, convinced us that the problem of capturing and automating common user tasks on the Web is interesting, and suggested to us an approach to its solution - Web Macros. Web Macros are programs that emulate user interaction with the Web. They can navigate on the Web, are authored implicitly by observing user actions, and are validated against results retrieved by the user. Web Macros can be communicated to other users by factoring out and substituting personal information. We identified the following set of requirements for Web Macros:

Automated hypertext navigation

To model a process of user interaction with the Web, Web Macros must emulate user actions: retrieve documents, and fill out and submit forms. Web Macros must be able to extract hyperlinks from the HTML content and follow them. Automated navigation steps may need to be scheduled on a timeline, to decrease load on the server(s) and avoid being perceived as a "denial of service" attack.

Maintaining the browsing context

While the HTTP protocol is connectionless, its extensions allow maintenance of state on the server. To support stateful transactions, Web Macros must store and transmit cookies and other information maintained in a connection-based protocol (https) such as user identity. Other types of navigation context, such as the identity of the HTTP client, might be needed (e.g., Microsoft requires that Windows updates are downloaded using Internet Explorer).

Authoring by example

Given a suitable programming language, Web Macros could be written explicitly by the user. However, this limits the use of Web Macros to programmers with good understanding of HTML and HTTP. We claim that creating Web Macros by observing user interaction with the Web is more promising since it supports all types of users and does not require writing and debugging code manually. If macros automating useful transactions on the Web can be implicitly created by a casual user, the information service providers are no longer required to support all or most potentially automatable tasks on the server.

For creating Web Macros by example, several models are possible:

- Explicitly marking the beginning and the end of a macro recording session.
- Letting the system infer the macro based on the current Web document as the desired result.
- Inferring macros from repeating navigation sequences - *auto-definition* [SK96].

Parameterization

Recording and replaying sequences of static URLs will not address many of the task automation issues. Determining which form fields should hardcoded into a macro, and which should be specified when a macro is being run is a serious challenge. Generality of a macro is balanced against reducing the amount of information user has to specify to use it. Deducing macro parameters from multiple examples is a common approach. We are investigating how user feedback when a macro is being replayed can help infer macro parameters.

Automated Validation

A Web macro is validated by executing it and comparing the retrieved document with the one explicitly retrieved by the user. The execution environment does not have to be the same as that of the browser. Bookmarks use the identity of the URL as the validation criterion. We are also using the following more complex criteria:

- Same HTML markup, but possibly different content. This is useful for macros retrieving a

document whose content can be different every time the macro is run. For example, the current "top story" in an on-line newspaper gets regularly updated by design.

- Same markup and content, possibly different embedded links. For example, a document can include links to advertisements, which are different every time it is retrieved, but do not change the "essential" content. A shopping cart falls into this category.

- "Sufficiently similar" markup and content. Similarity must be defined based on user feedback and domain- and site-specific rules.

A user should be able to select an initial comparison criterion, and provide feedback on the results of the comparison in the course of validation, allowing the system to adjust or change the criteria.

Substitutability of personal information

The second scenario in the previous section, the shareable shopping cart, shows that Web Macros must be available to a user different from the one who authored them. Two types of user-specific information need to be factored out:

- Private information, such as name, address, credit card number. These parameters should never be left unchanged when a macro is passed to another user. They must be substituted with data coming from the recipient user's profile; if these are not available, the system would prompt the user for relevant data.

- Non-private information. Examples of this are home airport for air travel reservations; while this type of data can be left unchanged when a macro is passed to another user, intelligently substituting it can increase the usability of Web Macros.

Related Work and Comparison

The explosion in the size and diversity of services and uses of the Web makes it an appealing ground for research in HCI, and the dynamic nature of the Web creates new problems for researchers to solve. Tools improving navigation, automating common tasks, and hiding idiosyncrasies of individual Web sites can lead to savings in time and cognitive effort. In this section, we review and compare several representative systems aiming at these goals. Features of the discussed systems are summarized in Table 1.

Bookmarks and favorites were conceived as a mechanism for maintaining personal information spaces for the Web of static documents. Bookmarks are a special case of Web Macros: single-step ones, with URL identity as the validation criteria, and without support for parameterization or using browsing context. Abrams et al [ABC98] identified management and recall problems arising when one's bookmark collection grew sufficiently large. [TW98] describes an approach to improve recall of saved pages by analyzing user browsing behavior and bookmark archive and showing results as clues for revisiting pages. Management and recall issues identified for large collections of bookmarks are likely to carry over to large collections of Web Macros.

The InfoBeans project [BD99], currently under development, will support building encapsulated information sources from the Web by demonstration. Services are encapsulated by explicitly specifying and connecting their inputs and outputs with a graphical interface. Resulting encapsulated services are displayed inside browser frames. Unlike Web Macros, InfoBeans do not support navigation, and

validation has to be done manually by the user. InfoBeans are not specifically designed for personalization and reuse.

Internet Scrapbook [SK98] is a system for merging Web information services, particularly those updated on a regular basis, into personalized Web pages. The reported user study shows that the system is robust to common changes in the markup structure of the underlying services. Internet Scrapbook uses sophisticated heuristics to achieve this robustness. Internet Scrapbook, unlike Web Macros, operates on static content, and does not support navigation for retrieval of dynamically generated documents. The design of Internet Scrapbook does not provide for parameterization of authored "scrapbook" pages.

	Bookmarks/ Favorites	InfoBeans	Internet Scrapbook	AgentSoft LiveAgent	Web Macros
Intended class of tasks	Creating and managing personalized views of the Web	Encapsulating heterogeneous Web information sources	Creating merged Web pages robust to changes in information source structure	Automating common interaction and information collection tasks on the Web	Automating common interaction tasks on the Web
Intended user audience	Casual user	Casual/inter-mediate user	Casual user	Casual user	Casual user
Creator and user distinct	No (but bookmarks can be published)	Possibly	No	Possibly	Possibly
Cognitive effort for use	Low (management problematic for large collection)	Medium	Low	Medium	Low-medium
Built-in "intelligence"	None	High	Some	Some	Some
Authoring method	Explicit	By demon-stration, with custom GUI for connecting services	By demon-stration	By observing explicitly initiated recording sessions and prompting user for parameters	By observing user interaction with the Web
Execution environment	Netscape browser or	JavaScript and DHTML-	Custom Web client	Recording in Netscape	Recording and

	Internet Explorer	capable browser		Communicator only with proprietary Java proxy	playback in any browser that supports proxies

Table 1: Comparison of Related Work

AgentSoft's LiveAgent [Kru97] is a proprietary system for automating repetitive browsing and data entry tasks on the Web. LiveAgent scripts that automate tasks are authored by demonstration, with the user explicitly initiating the recording session. LiveAgent does not try to infer constant and variable parameters of a script; rather, a user is prompted to make the decision for each field of a submitted form. To provide robustness against changes in HTML content and markup, LiveAgent uses HPDL, HTML Position Definition Language, to describe the links followed from pages retrieved during demonstration sessions. The user is prompted with a graphical dialog to specify HPDL for these links. LiveAgent scripts can be authored for shared use, but there is no provision for factoring out and substituting personal information.

Architecture of the Prototype

To validate our model and requirements for Web Macros, we built a prototype system for creation, validation and playback of scripts implementing a subset of functionality planned for Web Macros. The architecture of the system is shown on Figure 3.

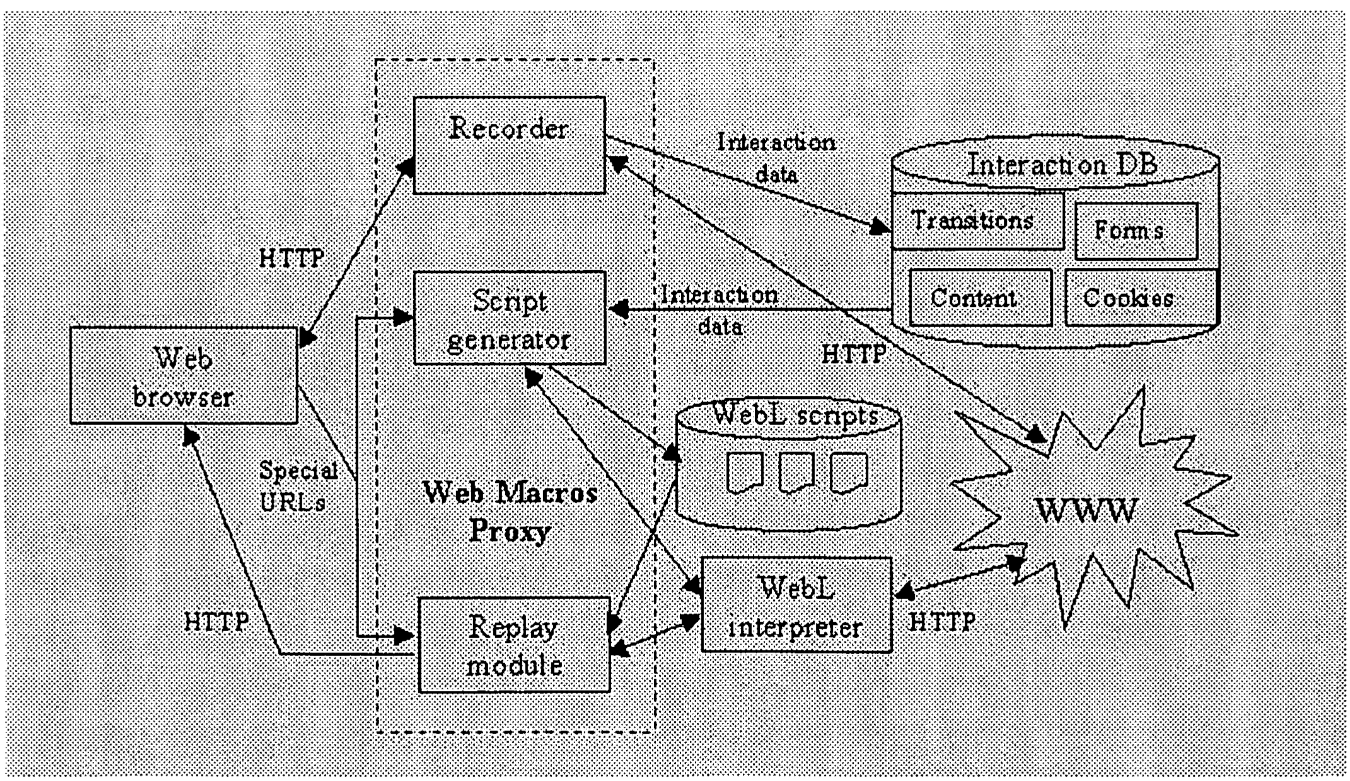

Figure 3: Architecture of the Web Macros System Prototype

The Web Macros prototype is designed as an HTTP proxy that sits between the Web and the browser. The recorder component parses the HTTP requests and replies and writes navigation actions to the interaction database. HTML markup of retrieved documents is also saved to validate generated macros. User can optionally request saving complete documents, or just checksums if exact match on content is desired. Submitted forms are parsed, and their values are saved. Cookies are captured and associated with documents for which these cookies were sent by the server. Each navigation action and document is marked with a timestamp, user identity and a session identifier, where a new session is started for each invocation of the browser. The macro recorder is built from a modified Muffin HTTP proxy [Boy98], which is written in Java and works with any standard Web browser that supports proxies, such as Netscape Navigator or Internet Explorer.

User interacts with the Web Macros system also through the standard browser interface, by navigating to special URLs starting with http://macros/. All such URLs are interpreted by the Web Macros system directly.

This current prototype uses the semi-explicit mode of macro creation. When a user wants to create a macro that would produce the current document, they would navigate to a special URL http://macros/record/ and enter the name of the new macro. Heuristic rules to measure markup and content similarity are used for macro validation. The script generator then reads interaction data from the database and attempts to generate a WebL[KM98] script that will, when executed, retrieve the desired document. This approach of creating a macro based on the desired result document is similar to the model of bookmarking Web pages.

The process of script creation and validation is iterative. First, the generator creates and evaluates a WebL script consisting of a single request to retrieve the current document. This could fail, because the server relies on previously established state that must be recreated. In case of failure, the generator creates and evaluates scripts with longer sequences of document requests and form submissions obtained from the interaction database. The system has rules for detecting login pages, and tries to emulate session tokens by detecting pages that contain hidden form fields. When finally a WebL script that generates the desired document is produced, it is saved in the database for future execution by the user.

The initial results of using the Web Macro prototype are positive. The first scenario described above - the bibliography database search - has been automated using the system. Author name is inferred as the only parameter, since the system recognizes that login name and password fields should have constant values. The bibliography search macros are shown in Figure 4.

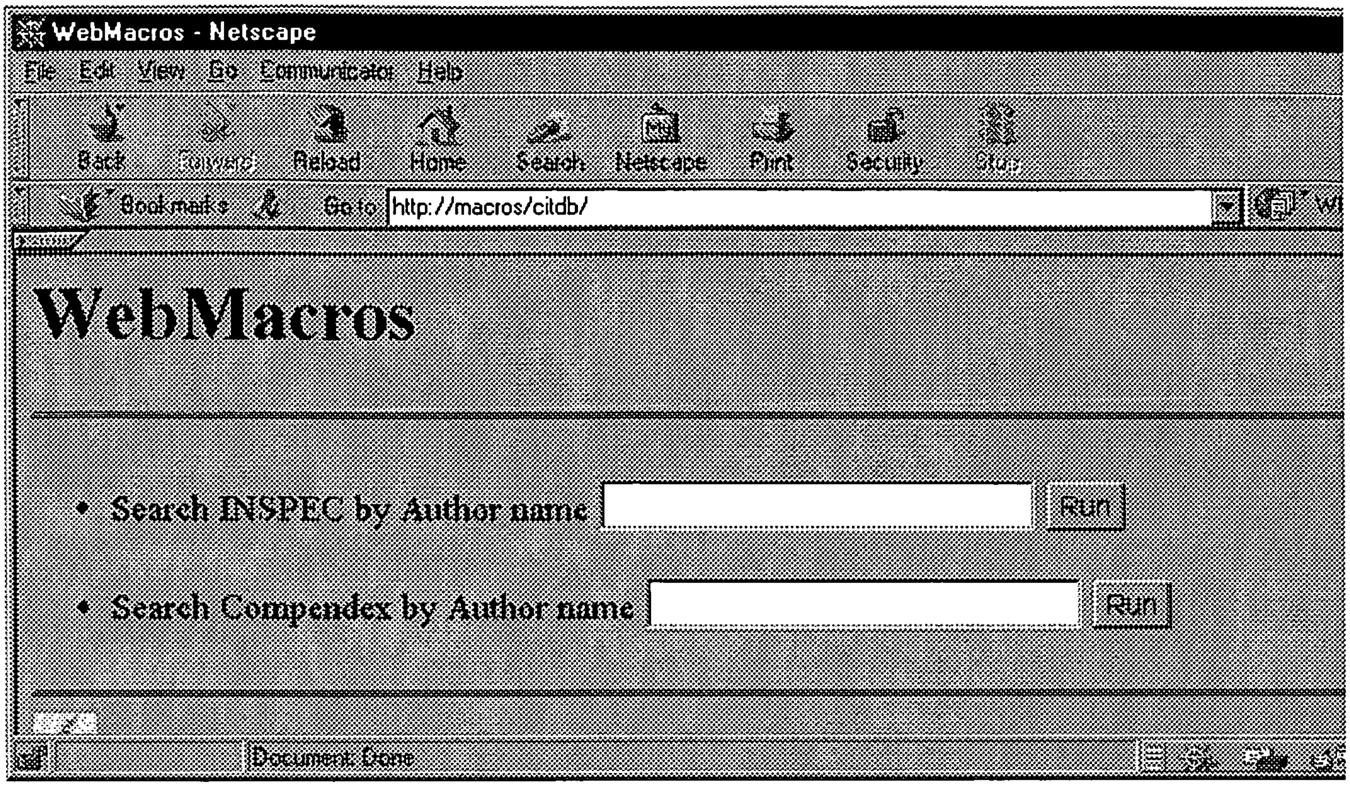

Figure 4: Interface to the Bibliography Search web macros

The second scenario - populating a shopping cart and passing it to another user - is currently being worked on. The challenge is to include with a macro a minimum set of cookies to successfully recreate the shopping cart contents, and at the same time not to disclose any unnecessary information.

Discussion and Future Work

We have identified some challenges for Web Macros and approaches to addressing them. We believe that Web Macros will be usable if they are generalizable - the issue that almost every PBE system attempts to address. We are investigating the patterns of Web macro usage by describing scenarios that can be automated, and analyzing the regularities in the markup structure of the retrieved documents. We are planning to use heuristics to identify "essential" fields that are changeable for a macro and that can be used to identify it to the user, a role similar to that served by a bookmark title.

Separating the Web interaction data from the actual macro scripts has several advantages. First, it gives the flexibility of using different scripting languages as appropriate. We are currently using WebL, since its Web navigation and HTML parsing primitives fit very well with our goals; however, it is also possible to generate Java code for wider compatibility. Second, a central repository of all Web interaction data allows this data to be shared among users, and potentially for users to share generated Web macros. Finally, a database (relational in our approach) representation makes it easier to construct graphical tools for extending the scripts, since reparsing of script code is not required.

Implementing and Evaluating Explicit Definition and Autodefinition of Macros

Currently the user is required to explicitly mark a document as the desired result of a macro to be inferred. This simple approach makes creating a macro similar to bookmarking a document, but limits

the generality of the generated macros. For example, in the bibliography database search, it is not possible to detect that the user was iterating over several databases. Explicitly specifying the start and end of the interaction to be generalized into a macro will provide more information to the macro inference engine.

Macro auto-definition represents an opposite approach, in the sense that user does not explicitly specify either macro results, or its start and end. Rather, the interaction history is mined for common navigation sequences and macros are inferred from those. This approach has been studied in an office application environment [SK96] but not on the Web.

Detecting Regularities in Document Structure

Many automatable Web tasks involve submitting different parameters on each iteration of a loop and integrating search results into a single document for presentation to the user. Search results are typically returned with a fixed header, a variable-length list of hits, followed by a fixed footer. Being able to detect this structure and use it for merging results of iterations will make macros more useful.

Obtaining More Information for Macro Inference

Our current proxy-based implementation limits the types of information that can be recorded and from which Web macros are generated. In particular, there is no data on how the user scans rendered HTML in the browser with the mouse cursor, or copies, pastes, and retypes data in the forms. An approach to this problem is to dynamically enhance retrieved pages with JavaScript code which will communicate events to the macro recording engine.

Using Playback for Specifying Macros

Currently, when a Web macro is run, users are presented with a single document produced by the last command in the macro. Another approach is to present each retrieved document and submitted form, augmented with macro navigation controls, to the user. The user will be able to step through a macro, and perhaps deviate from its execution path. This feedback can be used to more precisely specify the macro.

Handling Transactions with Side-effects

Transactions that cause side effects cannot be normally repeated during validation. Examples of those are submissions of forms that purchase merchandise, reserve lodging, etc. Reliable identification of transactions with side effects may require extensions to the HTML standard, specifying what, if any, side effects or classes or side effects a submitted form or a hyperlink has. We believe it will be useful to support *alternate* actions or URLs that do not cause side effects but have at least a part of the desired functionality.

Identifying More Scenarios and Conducting User Studies

Many of the scenarios of automatable tasks we are investigating were suggested to us by the members of the Collaborative Filtering, Databases and Multimedia research groups at the Computer Science department. We are planning an informal pilot study of the Web Macro prototype with graduate students in the department (who are expert computer users and developers), and a larger study which will recruit users of various skill levels.

The challenges of providing Web users with general, reusable, intelligent tools for automating common tasks are high. We have started the research and built a prototype implementation, and will strive for achieving the requirements, within the environment provided by the current Web technology, and by extending it. Besides a publicly available, easy-to-use Web task assistant, we plan to produce a set of recommendations for Internet standards committees, content creators, and browser and server software developers to help better support Web automation.

Acknowledgements

This work is supported by the NSF grant DBI-9723816. We are grateful to Brian Bailey, Jon Herlocker, and Paul Wagner for valuable comments. The Muffin HTTP proxy[Boy98] and WebL [KM98] provided extensive functionality we could build upon.

Bibliography

[ABC98] Abrams, D., Baecker, R., and Chignell, M. Information archiving with bookmarks: personal Web space construction and organization. *Proceedings of CHI '98, Conference on Human factors in computing systems*, 1998.

[Boy98] Boyns, M. Muffin - A World Wide Web Filtering System. Unpublished Master's Thesis, San Diego State University, 1998. http://muffin.doit.org/

[BD99] Bauer, M. and Dengler, D. InfoBeans-Configuration of Personalized Information Services. *Proceedings of ACM IUI 99.*

[KM98] Kistler, T. and Marais, H. WebL - A Programming Language for the Web. *Proceedings of WWW7 conference*, 1998.

[Kru97] Krulwich, Bruce. Automating the Internet: Agents as User Surrogates. *IEEE Internet Computing*, Volume 1, Number 4 (July/August 1997).

[SK96] Sugiura, A. and Koseki, Y. Simplifying macro definition in programming by demonstration. *Proceedings of the ACM Symposium on User Interface Software and Technology*, 1996.

[SK98] Sugiura, A. and Koseki, Y. Internet Scrapbook: Automating Web Browsing Tasks by Demonstration. *Proceedings of ACM Symposium on User Interface Software and Technology*, 1998.

[TW98] Takano, H. and Winograd, T. Dynamic Bookmarks for the WWW. Managing Personal Navigation Space by Analysis of Link Structure and User Behavior. Short paper in *Proceedings of ACM Hypertext'98.*

"Towards Web Macros: a Model and a Prototype System for Automating Common Tasks on the Web"

<- Back

Site Created: Dec. 12, 1998
Last Updated: June 10, 1999
Contact hfweb@nist.gov with corrections.